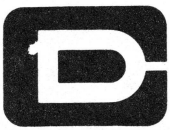
DOT

BUSINESS AND FUN WITH BASIC:

A Sampler of BASIC Programs for DOT™

BUSINESS AND FUN WITH BASIC:

A Sampler of BASIC Programs for the DOT™



COMPUTER DEVICES INC.
25 North Avenue
Burlington, MA 01803
Tel: (617) 273-1550

The information in this document is subject to change without notice and should not be construed as a commitment by Computer Devices, Inc. and, in the case of software authored by third parties, should not be construed as a commitment by such third party.

Software that may be described in this document is furnished under the Software Program License Agreement which is provided in its entirety below.

Computer Devices, Inc. and the author of software in the case of software authored by such third party, assumes no responsibility for the use or reliability of its software for any errors that may appear in this document.

COMPUTER DEVICES, INC.
PROGRAM LICENSE AGREEMENT

COMPUTER DEVICES, INC. ("LICENSOR") PROVIDES THE ENCLOSED PROGRAM (THE "PROGRAM") AND LICENSES ITS USE ONLY ON THE FOLLOWING TERMS AND CONDITIONS. YOU SHOULD CAREFULLY READ THE TERMS AND CONDITIONS BEFORE OPENING THIS PACKAGE. OPENING THIS PACKAGE INDICATES YOUR ACCEPTANCE OF THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THEM, YOU SHOULD PROMPTLY RETURN THE PACKAGE, UNOPENED, TO LICENSOR OR ITS AUTHORIZED DISTRIBUTOR, AND YOUR MONEY WILL BE REFUNDED.

1. COPYRIGHT.

The Program and its documentation are copyrighted by Licensor or its supplier. You may not use, copy, modify or transfer the Program, or any copy, modification or merged portion of the Program or documentation except as expressly authorized in this Agreement.

2. LICENSE.

You may use the Program only on a single machine. You may transfer the Program and the License from one machine to another, provided that you do not use the Program on more than one machine at a time. You may not make the Program available through a network of machines unless expressly authorized by Licensor in writing.

3. BACKUP COPIES.

If the Program is not marked "copy protected" you may make one machine readable copy of the Program for backup purposes in support of your use of the Program on the single machine. If the Program is marked "copy protected" and a backup copy is not included in this package, you may obtain one by completing the enclosed user registration card and returning it to Licensor with a request for a backup copy of the Program. You may not copy the Program or any of its documentation in printed form.

4. MODIFICATION.

You may not modify the Program or merge it into another program, except as may be expressly authorized in the user manual for the Program. Any portion of the Program that is merged into another program will continue to be subject to the terms and conditions of this Agreement. You must, as a condition of the License, reproduce and include the copyright notice on any copy, modification or merged portion of the Program.

5. ASSIGNMENT.

You may transfer the program and assign your rights under this agreement to another party if the other party agrees to accept the terms and conditions of this Agreement. If you transfer the Program, you must at the same time either transfer all copies thereof in printed or machine readable form to the same party or destroy any copies not transferred. If you transfer the Program, your license is automatically terminated.

6. TERM.

The License granted in this Agreement is effective until terminated. You may terminate it at any time by destroying the Program together with all copies, modifications and merged portions and notifying Licensor. The License granted in this Agreement

will also terminate if you fail to comply with any of the terms and conditions of this Agreement. Upon such termination, you must destroy the Program together with all copies, modifications and merged portions thereof.

7. LIMITED WARRANTY.

Licensor warrants only (a) that the media on which the Program is furnished will be free from defects in materials and workmanship under normal use for a period of 90 days from the date of delivery to you as evidenced by a copy of your receipt.

ALL SOFTWARE PRODUCTS LICENSED BY COMPUTER DEVICES, INC. ARE PROVIDED AS IS WITHOUT ANY WARRANTY OF ANY KIND EITHER EXPRESS OR IMPLIED.

YOU ASSUME FULL RESPONSIBILITY FOR THE SELECTION OF THE PROGRAM TO ACHIEVE YOUR INTENDED PURPOSES, FOR THE PROPER INSTALLATION AND USE OF THE PROGRAM AND FOR VERIFYING THE RESULTS OBTAINED FROM USE OF THE PROGRAM. LICENSOR DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE PROGRAM WILL MEET YOUR REQUIREMENTS, THAT THE PROGRAM IS FIT FOR ANY PARTICULAR PURPOSE OR THAT THE OPERATION OF THE PROGRAM WILL BE UNINTERRUPTED OR ERROR FREE.

Should you discover a defect in the recording medium within the 90 day warranty period, then return the defective medium to Licensor at the address indicated below. Include a copy of your receipt or other proof that the 90 day warranty period has not expired. Licensor will, upon verification of the defect either replace the defective copy or refund the amount you paid.

THIS WARRANTY IS IN LIEU OF ALL OTHER WARRANTIES, WHETHER ORAL OR WRITTEN, EXPRESS OR IMPLIED. ANY IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO 90 DAYS FROM THE DATE OF DELIVERY TO YOU.

Some states do not allow the exclusion of implied warranties, so the above exclusion may not apply to you. This limited warranty gives you specific legal rights and you may also have other rights which vary from state to state.

8. LIMITATION OF REMEDIES.

YOUR EXCLUSIVE REMEDY AND LICENSOR'S ENTIRE LIABILITY SHALL BE AS SET FORTH ABOVE. IN NO EVENT WILL LICENSOR BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM OR THE BREACH OF ANY EXPRESS OR IMPLIED WARRANTY EVEN IF LICENSOR OR ITS DEALER OR DISTRIBUTOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

9. GENERAL.

You may not sublicense, assign or transfer the License or the Programs except as expressly provided in this Agreement. Any attempt to do so, shall void the License.

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU FURTHER AGREE THAT IT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND LICENSOR AND SUPERSEDES ANY PROPOSAL OR PRIOR AGREEMENT OR ANY OTHER COMMUNICATIONS BETWEEN YOU AND LICENSOR RELATING TO THE USE OF THE PROGRAM.

If you have any questions regarding this Agreement, you may contact Licensor by writing or calling Licensor at the following address:

Computer Devices, Inc.
25 North Avenue
Burlington, Massachusetts 01803
Phone: 617-273-1550

0008-14613/1500
Copyright 1983, Computer Devices, Inc.
All rights reserved.
Printed in U.S.A.

GUIDE TO DOT DOCUMENTATION

To learn how to perform specific tasks with your DOT, read the following DOT documents in the order recommended.

TASK	MEET YOUR DOT™: An Introduction to the DOT Computer (0008-14612)	SOFTWARE DECISIONS: The DOT™ Guide to Software Selection (0008-14611)	YOUR DOT™ FILE CABINET: File Editing and Housekeeping on the DOT (0008-14615)	BUSINESS AND FUN WITH BASIC: A Sampler of BASIC Programs for the DOT™ (0008-14613)
Care for Equipment	1st			
Choose Software		1st		
Create Files	1st		2nd	
Edit Files	1st		2nd	
Manage Files	1st		2nd	
Run BASIC	1st			2nd
Run Programs	1st			2nd
Set Up the DOT	1st			
Start the DOT	1st		2nd	
Troubleshoot the DOT	1st			
Use the DOT as a Terminal	1st			

CONTENTS

Section 1
INTRODUCTION 1-1

Section 2
USING BASIC 2-1

Loading BASIC 2-1
Running a Simple Program 2-4
Correcting Typing Mistakes 2-5
Listing Your Program 2-5
Saving Your Program 2-5
Reloading Your "Saved" Program 2-6
Creating a New Program 2-6
Exiting from BASIC 2-7

Section 3
WRITING SIMPLE BASIC PROGRAMS 3-1

Understanding BASIC Concepts 3-1
- Understand BASIC Keywords 3-1
- Start Small 3-2
- Include Remarks 3-3
- Take One Step at a Time 3-3
- Use Abbreviations as Variable Names 3-4
- Follow BASIC Math Rules 3-5

Programming Step-by-Step 3-6
- Begin With the Calculations 3-6
- Add the Data 3-8
- Calculate and Print the Output 3-9

Section 4
SAMPLE BASIC PROGRAMS 4-1

Return on Investment 4-2
- Program 4-2
- Sample Run 4-2

Depreciation 4-3
- Program 4-3
- Sample Run 4-3

Standard Deviation 4-4
- Program 4-4
- Sample Run 4-4

Decision Maker 4-5
Program 4-5
Sample Run 4-6
Over-and-Under Dice Game 4-7
Program 4-7
Sample Run 4-7
Factors of a Number 4-8
Program 4-8
Sample Run 4-8
Number Averaging 4-9
Program 4-9
Sample Run 4-9
Wallpaper Cost Estimating 4-10
Program 4-10
Sample Run 4-11
Hangman 4-12
Program 4-12
Sample Run 4-13
Loan Payments 4-15
Program 4-15
Sample Run 4-15
Loan Payoff 4-16
Program 4-16
Sample Run 4-16
Costing Real Estate Points 4-17
Program 4-17
Sample Run 4-17
Age in Days 4-18
Program 4-18
Sample Run 4-19
Salvage Value 4-20
Program 4-20
Sample Run 4-20
Effective Interest Rate 4-21
Program 4-21
Sample Run 4-21

Section 5
AVAILABLE BASIC TEXTBOOKS 5-1

Appendix A
GLOSSARY A-1

Section 1
INTRODUCTION

Section 1
INTRODUCTION

Programs enable you to communicate instructions to your computer. To make the best use of your DOT™ Personal Computer, you need programs that are designed for your particular applications. Many programs are available for common applications such as accounting, financial planning, job scheduling, and word processing. However, you may have special requirements that these programs do not address.

If you cannot find the right program for your application, you can adapt your system to an existing program or you can write a new program. With the DOT Personal Computer, you can quickly learn to write your own programs in BASIC (Beginner's All-purpose Symbolic Instruction Code).

To simplify programming, BASIC uses English keywords (or their abbreviations) as instructions which tell the computer what to do. BASIC turns these keywords into the electronic signals that the DOT understands.

This Sampler of BASIC programs gives you a quick look at the BASIC programming language in operation. By following the instructions in Section 2, "Using BASIC", you will learn how to enter and run a simple BASIC program.

Section 3, "Writing Simple BASIC Programs", explains some rules of the BASIC language. This section won't make you an expert programmer, but it will familiarize you with BASIC by explaining how to program your DOT to work like a powerful programmable calculator.

Section 4, "Sample BASIC Programs", contains 15 simple programs which you can enter and run. Each program includes a short description and sample output to help you get started using BASIC.

DOT is a trademark of Computer Devices, Inc.

To learn more about BASIC, see the list of reference books in Section 5, "Available BASIC Textbooks". These books provide details on programming in BASIC.

Before you read Section 2, "Using BASIC", you should be comfortable with the operation of the DOT computer. Make sure you have read **Meet Your DOT: An Introduction to the DOT Computer** before continuing with this manual.

Section 2
USING BASIC

Section 2
USING BASIC

The DOT computer supports many different programming languages of which BASIC is the easiest to use. When talking about BASIC, note that "the BASIC language program" is different from "an application program written in the BASIC language". The BASIC language program is the master program that defines the rules and vocabulary of the BASIC language. An application program written in BASIC consists of instructions that use the BASIC language vocabulary and rules for accomplishing specific tasks.

> The BASIC language program for the DOT is Microsoft® BASIC. Microsoft BASIC has been the industry-standard BASIC for small computer systems since 1977.

Application programs are much easier to write in BASIC than in other programming languages because BASIC instructions are English-like. Each time the DOT reads an instruction in your BASIC application program, it translates that information into a series of machine-language instructions for the computer to perform.

LOADING BASIC

The BASIC language program is one of the programs that you receive with your DOT. You need to load the BASIC language program into the DOT before you can use any application programs written in BASIC. To load BASIC into the DOT, follow these steps:

1. Put your working copy of the system diskette in the disk drive (the left-hand drive on a two-drive system). Be sure to open the sliding door on the diskette before you put it in the disk drive.

Note: If you do not know what a "working copy" of the system diskette is, refer to **Meet Your DOT: An Introduction to the DOT Computer**.

Microsoft is a registered trademark of Microsoft Corporation.

2. To bring the operating system program into the DOT's memory, first press the RESET button on the left rear of the DOT. (In computer jargon, this step is called "booting the system".) Another way to start the system is to press and hold down the **[CTRL]** key and the **[ALT]** key while you press the **[CLR]** key. When you use either method, the screen displays the following information:

```
                              D O T

                COPYRIGHT (C) 1983 COMPUTER DEVICES, INC.

                        1. LOAD FROM LEFT DRIVE (A)
                        2. LOAD FROM RIGHT DRIVE (B)
                        3. USE AS TERMINAL
                        4. RUN EXTENDED DIAGNOSTICS

Enter the number for the desired function > _
```

3. Press the number **1** key to load the MS-DOS operating system into memory. At this point, the DOT displays the following message:

```
MS-DOS version 1.25
Copyright 1981,82,83 Microsoft, Inc.
Io System - Version 02.00.00

Command v. 1.17.1
Current date is Tue  1-01-1983
Enter new date: _
```

4. To accept the displayed date, simply press **[RETN]**. To change the date, enter the current date and press **[RETN]**. The following information appears on the screen:

```
Current time is 0:05:00.00
Enter new time: _
```

5. To accept the displayed time, press **[RETN]**. To change the time, enter the correct time and press **[RETN]**. The DOT displays the system prompt:

   A:

6. To place the BASIC language program into the DOT's memory, type the word **MBASIC,** in uppercase or lowercase letters, and press the **[RETN]** key. A message similar to the following appears on the screen:

```
BASIC-86 Rev. 5.27
[MS-DOS Version]
Copyright 1977-1982 (C) by Microsoft
Created:  6-Nov-82
62261 Bytes free
Ok
```

The number of bytes free may vary for different systems; you need not be concerned with this information now. This message simply tells you that the BASIC language program is loaded and ready to use. Now the DOT is waiting for you to type the contents of a BASIC application program at the keyboard or to load a BASIC application program from a disk.

RUNNING A SIMPLE PROGRAM

To see how easy BASIC is to understand, look at this short program:

```
PRINT  3 + 4
```

This program consists of the keyword PRINT, the addition sign, and the numbers 3 and 4 that serve as data. Type this line at the keyboard and press the **[RETN]** key to run the program. The number 7 and the word Ok should print on the screen.

```
PRINT  3 + 4 [RETN]
7
Ok
```

When you create programs that you want to save on disk, you must type a line number at the beginning of each line. Line numbers tell BASIC the order in which to record the lines of your program. For example, if you type lines 10 and 20, and then type line 15, BASIC inserts line 15 between lines 10 and 20. A program can operate without line numbers, but you will not be able to save it.

Now type this two-line program. (Note that the characters that you type appear in bold type.) Be sure to include spaces where shown.

```
10 X = 3 + 4 [RETN]
20 PRINT  X [RETN]
```

To run programs that include more than one line, you type the word **RUN** on a line by itself and press the **[RETN]** key. Type **RUN** and press **[RETN]**. Your DOT displays the number 7 followed by the word Ok:

```
RUN
7
Ok
```

This program produces the same result as the previous program. The only difference between these programs is that the two-line program breaks the task down into two steps: addition and printing.

CORRECTING TYPING MISTAKES

If you make a mistake while typing your program, press the **[BACK SPACE]** key to erase errors within the line you are typing. After you press the **[RETN]** key, BASIC enters that line into memory.

If you see a mistake in a line after you press the **[RETN]** key, correct that line by retyping it with the same line number. BASIC remembers only the most recent version of any line that you enter. To erase a complete line, type its line number with no text after it and press the **[RETN]** key.

LISTING YOUR PROGRAM

You can view your program after you enter it by typing **LIST.** After you type this command, the DOT displays your program beginning with the first line and ending with the last. If you want to view only lines 10 through 50, you can enter **LIST 10-50** and the DOT will display the selected lines. Alternately, you can list your program on the DOT printer by typing **LLIST.**

SAVING YOUR PROGRAM

After you verify that your program works, you should save it on a diskette so that you can use it again later without retyping each line at the keyboard. When you tell the DOT to save your program, it copies the information from the working memory to the magnetic diskette.

You should save your BASIC application programs on a separate diskette from the system diskette. First, take the system diskette out of the disk drive. Then label a new, formatted diskette BASIC PROGRAMS and put it in the disk drive (the left-hand drive on a two-drive system). (If you do not know how to format a new diskette, refer to **Meet Your DOT: An Introduction to the**

DOT Computer.) Note that you can remove the system diskette from the drive any time after you load the BASIC language program into the DOT and the start-up BASIC message appears.

With a BASIC PROGRAMS diskette in the disk drive, save the two-line program as follows.

Type **SAVE "TEST"** and press the **[RETN]** key. You must give your program a name, in this case TEST, to store it on a diskette. Program names can consist of no more than 8 characters and cannot contain spaces or punctuation. Note that you must type quotation marks before and after the program name. BASIC responds by storing your program under the name TEST and printing Ok.

Now you have a copy of the program on a diskette. At this point, you can still execute (or "run") the program by typing **RUN**, because it is still in the DOT memory. The program leaves the DOT memory only when you load another program, create a new program, or turn off the DOT.

RELOADING YOUR "SAVED" PROGRAM

To reload a program that you stored on a diskette, simply type **LOAD** and the program name in quotation marks. Then press the **[RETN]** key. When you reload a program from a diskette, the DOT automatically clears any previously loaded BASIC program from memory.

Reload the program you saved earlier by typing **LOAD "TEST"** and pressing the **[RETN]** key. BASIC responds by loading your program and printing Ok.

LOAD "TEST" [RETN]
Ok

Now you can type **RUN [RETN]** to tell the DOT to execute the program in memory.

CREATING A NEW PROGRAM

Before you can create another program at the keyboard, you have to clear the computer's memory. In BASIC, the word NEW clears any program from the computer's memory. Type **NEW** and press the **[RETN]** key. The DOT responds by displaying the word Ok. Now you can enter a new program at the keyboard.

EXITING FROM BASIC

When you are finished using BASIC, perform the following steps to return to the system prompt.

1. If you haven't already saved your BASIC application program, decide if you want to save a copy of your program before you exit from BASIC. If you want to save your program on a diskette, follow the instructions under "Saving Your Program" earlier in this section.

2. To exit from BASIC, type the word **SYSTEM** on a line by itself and press the **[RETN]** key.

   SYSTEM [RETN]

   The system prompt, A:, appears on the screen.

   A:

Section 3
WRITING SIMPLE BASIC PROGRAMS

Section 3
WRITING SIMPLE BASIC PROGRAMS

There are two ways to prepare a program for execution: you can load it from a diskette or you can enter it at the keyboard. This section first explains some simple BASIC language concepts. Then it shows you how to write and enter a program that makes your DOT work like a powerful programmable calculator.

UNDERSTANDING BASIC CONCEPTS

To write a BASIC program, you need to understand some simple BASIC concepts such as keywords, remarks, abbreviations, and BASIC math rules.

Understand BASIC Keywords

BASIC keywords are the English-like instruction words that comprise a BASIC program. Consult a textbook on BASIC for a complete list of the valid keywords. Here are some BASIC keywords to help you understand the programs in this Sampler:

Keyword	Description
PRINT	Produces the output on the screen.
LPRINT	Directs the output to the printer.
INPUT	Prints a question mark on the screen. This question mark indicates that the program requires data from the user before it can continue.
GOTO	Indicates which line in the program to read next, rather than the next numerical line.
LIST	Displays the entire program or specified lines of the program on the screen.

continued on next page

Keyword	Description
REM	Enables you to enter comments that describe or annotate the program.
FOR	Begins a program loop and specifies how many times to perform the loop operations.
NEXT	Sends program execution back to the matching FOR statement.
STEP	Determines how much to increment or decrement the FOR-NEXT loop variable for each pass through the loop.
IF	Tests a variable for a specified condition.
THEN	Describes the action to take if the condition in an IF statement is met.
ELSE	Describes the action to take if the condition in an IF statement is not met.
RANDOMIZE	Accepts a random number "seed" to begin generating random numbers with the RND keyword.
RND	Produces a random number.
GOSUB	Sends program execution to a subroutine for more processing.
RETURN	Returns from a subroutine to the statement following the GOSUB statement.
END	Stops the program.
LLIST	Lists your program at the printer.

A colon (:) separates different commands on the same line.

Keywords are the building blocks for a BASIC program. They enable you, the programmer, to tell the computer what to do.

Start Small

When writing a program, begin with a small portion of the program and add to it. This way, you can test each stage of your program. For example, enter the following lines which form the beginning of an Interest Calculator program:

10 REM Decimal equivalent of 14.6 annual interest rate (AR) [RETN]
20 REM Enter the annual interest rate [RETN]
30 AR = 14.6 [RETN]
40 REM Divide by 100 to get a decimal rate (DAR) [RETN]
50 DAR = AR/100 [RETN]
60 REM Print the results [RETN]
70 PRINT DAR [RETN]

These lines form a small program themselves. Test this first portion of the Interest Calculator program by typing **RUN** and pressing the **[RETN]** key. The correct result is .146.

RUN [RETN]
.146
Ok

If you did not get .146, correct any errors as explained in Section 2 under "Correcting Typing Mistakes" and rerun the program.

Include Remarks

REM stands for "remark". Remark lines contain comments that explain what the program is doing. They are tools to help you, the programmer. BASIC produces the same results whether you include remark lines or not. However, programs that include remarks are easier to read and understand.

Take One Step At a Time

Each line in the first portion of the Interest Calculator program (except the remark statements) performs one calculation step. Now expand this program by adding two more lines: one line to explain the next calculation and one line to calculate the interest rate for 1 month. Insert these new lines between lines 50 and 60 by assigning them line numbers between 51 and 59.

Type the following two lines:

52 REM Divide by 12 = monthly rates[RETN]
54 DMR = DAR/12 [RETN]

BASIC inserts the new lines and processes them in numerical order.

Finish this Interest Calculator program by replacing line 70 with a new line 70. Type:

70 PRINT DMR [RETN]

Type **LIST** and press the **[RETN]** key. Your program appears on the screen with the two new lines between lines 50 and 60, and the new instruction in line 70.

```
LIST
10 REM Interest Rate Conversions
20 REM Enter the annual interest rate (AR)
30 AR = 14.6
40 REM Divide by 100 = decimal rate (DAR)
50 DAR = AR/100
52 REM Divide by 12 = monthly rate
54 DMR = DAR/12
60 REM Print the results
70 PRINT DMR
Ok
```

Your finished program calculates and prints the monthly interest rate at 14.6% annual interest.

To see the result, type **RUN** and press **[RETN]**. The screen displays:

RUN [RETN]
1.216667E − 02
Ok

> Note that the E − 02 at the end of this number indicates to move the decimal point left two places to get .01216667, the correct result. E + 5 at the end of a number means to move the decimal point right five places to get the complete number.

Use Abbreviations as Variable Names

In the Interest Calculator program, the names that represent numeric values are called "variables" because they may vary each time you use the program. Abbreviating these names to just a few characters minimizes the number of keystrokes you need to type.

For example, AR stands for annual rate, DAR stands for decimal annual rate, and DMR stands for decimal monthly rate. You can also choose other abbreviations as variable names, such as ANRATE for annual rate, as long as they follow these three rules.

- Variable names can be no more than 40 characters.
- Variable names cannot contain spaces.
- Variable names cannot be the same as a BASIC keyword. For example, you cannot abbreviate "interest" as INT because INT is the BASIC keyword that converts a mixed number into an integer (such as changing 4.5 into 4).

Note that when you make AR equal 14.6 in line 30, the program automatically replaces AR with 14.6 whenever it encounters AR in an equation.

Variables make it easy to write one program that can process different data. For example, simply changing line 30 to AR = 9.8 calculates the monthly interest based on a 9.8% annual rate.

Follow BASIC Math Rules

There are many ways to perform a single operation in BASIC. In the Interest Calculator program, line 30 entered the annual interest rate, line 50 converted a percentage interest rate into a decimal interest rate, line 54 calculated the monthly interest rate, and line 70 printed the results. You can also combine these steps into one line that reads PRINT 14.6/1200.

Save the Interest Calculator program as explained in Section 2, "Using BASIC". Then clear the computer memory by typing **NEW** and pressing the **[RETN]** key. Now type **PRINT 14.6/1200** and press the **[RETN]** key. This one-line program produces the same result as the longer Interest Calculator program.

If you put one simple calculation on each line (as in the nine-line Interest Calculator program), the order in which BASIC performs arithmetic doesn't matter. However, this approach sometimes is inconvenient and requires several lines. To write a complex equation on one line in BASIC, observe the following rules:

1. BASIC calculates functions (such as square root or cosines) first.
2. BASIC performs squaring, cubing, quadrupling, and raising to other powers second.
3. BASIC performs multiplication and division third.

4. BASIC performs addition and subtraction fourth.
5. You can override any of these rules by using parentheses to isolate the operation you want to perform first.

Suppose you want to add 4 and 6 and then divide the total by 5. Type **PRINT 4+6/5** and press the **[RETN]** key. You get 5.2, which is wrong. BASIC divided 6 by 5 first and then added 4 to the result.

To tell BASIC to add first, simply enclose the addition part of the equation in parentheses. Type **PRINT (4+6)/5** and press the **[RETN]** key. The correct answer, 2, displays on the screen.

```
PRINT (4+6)/5 [RETN]
2
Ok
```

It is a good rule to use parentheses to designate which calculations BASIC is to perform first.

PROGRAMMING STEP-BY-STEP

A program consists of three distinct parts: the calculations, the input, and the output.

Begin With the Calculations

When you begin writing a program, start with the calculations. Work the program through on paper at least once before you type the program at your keyboard. As you calculate by hand, write a description of each step so that you can convert it to BASIC.

> Books specializing in calculator math are good sources for equations that fit many applications. You can turn these equations into BASIC lines and then build your program around this core.

For practice, begin with this equation for computing the monthly payments on a mortgage:

$$\text{Payment} = \frac{\text{Balance Owed}}{\left[\frac{1-(1+\text{Monthly Interest})^{-(\text{Number of Payments})}}{\text{Monthly Interest}}\right]}$$

To convert this equation into a program, first choose names for the variable elements. In this case, the variables are the balance owed, the monthly interest, the number of payments, and the payment amount. Use B, M, N, and P as names for these variables.

Second, convert the equation to its BASIC equivalent. To do this, rewrite the equation from the inside out. Use parentheses to be sure that BASIC performs the calculations in the order you would use with a calculator. For example, the following steps convert the monthly payments equation into its BASIC equivalent.

1. Find the part of the equation to perform first. In this example, the first calculation is (1+Monthly Interest). Rewrite this part as (1+M).

2. Next, raise the (1+M) to the negative power of the number of payments. Adding this second part changes the BASIC formula to (1+M)^(−N), where the up arrow signifies a quantity being raised to a power.

3. Now take the value just calculated and subtract it from one. The BASIC formula becomes 1−(1+M)^(−N).

4. The brackets in the original equation indicate that the next operation is dividing the current result by M, the monthly interest. Simply surround the BASIC formula from the previous step with another pair of parentheses, and divide by M. The new BASIC formula is:

   (1−(1+M)^(−N))/M

5. To divide the balance owed by the result of the last step, add another pair of parentheses and divide to get:

   B/((1−(1+M)^(−N))/M)

6. Now you have converted the right side of the equation into its BASIC equivalent. Last, include the variable name for the resulting payment amount on the left side by adding P =. The final BASIC formula is:

   P = B/((1−(1+M)^(−N))/M)

This BASIC formula does not look like the original equation and it certainly does not look any simpler. You have converted a complicated equation from one language to another. You don't have to understand the formula—you only have to understand

the order in which you must perform the calculations and the symbols that you use to express them. Set the finished BASIC formula aside while you write the next part of the program.

Add the Data

After developing your BASIC formula, you supply the input data. First, decide which data will remain fixed and which data will change every time you run the program.

Information such as conversion factors or fixed dimensions should be included as a permanent part of your program. This way, you don't have to tell your program that there are 12 months in a year or 144 units in a gross.

You can include fixed numbers in a program in two ways. The Interest Calculator program in this section shows you how to use a variable name to represent a constant number. The line AR = 14.6 enables you to use AR in place of the number 14.6. This technique, with variable names in place of numbers, makes the program easier to understand.

The BASIC monthly payment formula just developed shows how to include constant numbers directly in the BASIC line. That is, the number 1 must always be in the formula to make it work.

Variable data such as how much money you plan to deposit should be part of the data you enter each time you run the program. The INPUT keyword enables you to enter variable data. For example, to enter the numbers for the monthly mortgage payment program you are developing, type the following BASIC lines. Note that these three lines skip line number 20 so that you can insert a line between them in the next step.

```
10 INPUT "Monthly interest";M [RETN]
30 INPUT "Number of payments";N [RETN]
40 INPUT "Balance owed";B [RETN]
```

When you run the finished program, BASIC prints the text within quotation marks to tell you what you are entering. For example, when you run the program, the message "Monthly interest?" appears on the screen and the program waits for your input.

The variable names in these lines, such as M in line 10, tell BASIC to set the variable equal to the number you enter in response to the question, such as "Monthly interest?".

Calculate and Print the Output

Now insert a calculation in line 20 to convert the interest rate to a decimal, enter the BASIC formula onto line 50 to calculate the payment amount, and add a PRINT statement on line 60 to display the results. The program is almost complete:

20 M=M/100 [RETN]
50 P = B/((1−(1+M)/\(−N))/M) [RETN]
60 PRINT P [RETN]

Last, insert an instruction that will rerun the program without your typing RUN again and again. This way, you can easily reuse the program trying different input numbers to see which payment schedule best fits your plans.

You can add the following two lines to the end of any program to make it simple to choose to rerun or end the program.

50000 INPUT "Try again (y or n)"; ANS$ [RETN]
50010 IF ANS$ = "y" OR ANS$ = "Y" THEN 10 [RETN]

The large line numbers ensure that these statements will be at the end of your program. Note that, in this example, you want the program to begin again at line 10. If you want your program to return to a different line number, change the 10 after the keyword THEN in line 50010 to the number of the desired line.

When you type **LIST** and press **[RETN]**, your finished program should look like this:

```
LIST
10 INPUT "Monthly interest";M
20 M=M/100
30 INPUT "Number of payments";N
40 INPUT "Balance owed";B
50 P = B/((1-(1+M) ^ (-N))/M)
60 PRINT P
50000 INPUT "Try again (y or n)";ANS$
50010 IF ANS$ = "y" OR ANS$ = "Y" THEN 10
```

To run your program, type **RUN** and press the **[RETN]** key. Test your program by typing the appropriate numbers in response to

the questions. And, as always, remember to save your program as soon as it works to your satisfaction, as described in Section 2, "Using BASIC".

Section 4
SAMPLE BASIC PROGRAMS

Section 4
SAMPLE BASIC PROGRAMS

This section contains 15 short programs that you can enter and run on your DOT. Most of the programs perform calculations frequently used in business. You can execute the programs to calculate values such as return on investment, real estate points, salvage value, effective interest rates, depreciation, averages, and standard deviation.

You could also use a pocket calculator to determine the values that these programs calculate. However, using your DOT takes less time, especially when you want to recalculate the result using different numbers.

A few of the programs in this section are for fun. The Decision Maker program organizes your thoughts to help you make difficult decisions. The Age in Days program calculates the number of days since your birthdate. The Hangman program tests your vocabulary of four-letter words, and the Over-and-Under Dice Game program lets you gamble imaginary money.

For each program in this section there are three kinds of information: a description, the program with brief explanations in the right column, and a sample run. The description provides the purpose of the program and the data it requires.

The program itself appears just as you should type it (character by character and space by space) on the keyboard. **[RETN]** instructs you to press the carriage return key (do not type the letters RETN). At the right of the program are brief explanations of the important lines, to help you understand how the program was written. Caution: Do not type the explanations.

The sample run of each program illustrates the program in use. In the sample runs, the information that you type on the keyboard appears in bold type.

Before you enter a program, remember to load the BASIC language into the DOT as described in Section 2, "Using BASIC". Then type the program (including all spaces) just as shown in this section. Do not type the explanations at the right of the program.

After you enter a program and verify that it works, you can save it on a diskette as described in Section 2. Saving the program eliminates the task of retyping the program each time you use it.

RETURN ON INVESTMENT

This return-on-investment program calculates the final value of an investment. When you enter the interest rate, the number of compounding periods per year, the amount invested, and the number of years invested, the program calculates the final value.

Program

```
10 PRINT:INPUT "Annual interest rate";AR [RETN]

20 DAR = AR/100 [RETN]

30 INPUT "No. of compounding periods per year";CP [RETN]

40 INPUT "Amount invested";AM [RETN]
50 INPUT "No. of years invested";YRS [RETN]

60 V = AM*((1+(DAR/CP))/\(YRS*CP)) [RETN]
70 PRINT:PRINT "Final value = $";V [RETN]

80 INPUT "Another try (y or n)";A$ [RETN]
90 IF A$="y" OR A$="Y" THEN 10 [RETN]
```

Prints a blank line and then records the interest percentage.
Converts the percentage to a decimal value.
Records the number of compounding periods.
Records the amount invested.
Records the number of years invested.
Computes the investment value.
Prints a blank line and then prints the results.
Asks if the session is over.
If the session is not over, restarts at line 10.

Sample Run

```
run [RETN]

Annual interest rate? 14 [RETN]
No. of compounding periods per year? 12 [RETN]
Amount invested? 5000 [RETN]
No. of years invested? 5 [RETN]

Final value = $ 10028.03
Another try (y or n)? n [RETN]
Ok
```

DEPRECIATION

This depreciation program calculates the amount of depreciation of goods over a period of time. When you enter the depreciation rate, the amount invested, and the number of years over which the goods are to be depreciated, the program calculates the amount depreciated.

Program

```
10 PRINT:INPUT "Annual depreciation rate %";DR [RETN]
```
Prints a blank line and then records the depreciation rate.

```
20 DDR = DR/100 [RETN]
```
Converts the percentage to a decimal value.

```
30 INPUT "Amount invested";AM [RETN]
```
Records the amount invested.

```
40 INPUT "Number of years depreciated";YRS [RETN]
```
Records the number of years depreciated.

```
50 TOTDEP = 0 [RETN]
```
Initializes total depreciation.

```
60 PRINT:PRINT "Year","Depreciation" [RETN]
```
Prints column headings.

```
70 FOR X=1 TO YRS [RETN]
```
Performs one calculation for each year.

```
80 DEP = AM*DDR*((1-DDR)^(X-1)) [RETN]
```
Calculates the depreciation for one year.

```
90 PRINT X, DEP [RETN]
```
Prints information for each year.

```
100 TOTDEP = TOTDEP+DEP [RETN]
```
Sums the yearly depreciation amounts.

```
110 NEXT X [RETN]
```
Returns to line 70 until all years are depreciated.

```
120 PRINT:PRINT "Tot. amt. depreciated = $";TOTDEP [RETN]
```
Prints a blank line and then prints the results.

```
130 INPUT "Try another (y or n)";A$ [RETN]
```
Asks if the session is over.

```
140 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
If the session is not over, restarts at line 10.

Sample Run

run [RETN]

Annual depreciation rate %? **18 [RETN]**
Amount invested? **9000 [RETN]**
Number of years depreciated? **10 [RETN]**

Year	Depreciation
1	1620
2	1328.4
3	1089.288
4	893.2162
5	732.4373
6	600.5986
7	492.4908
8	403.8425
9	331.1508
10	271.5437

Tot. amt. depreciated = $ 7762.968
Try another (y or n)? **n [RETN]**
Ok

STANDARD DEVIATION

Standard deviation, also called the root mean square deviation, is the amount of variation in a group of numbers. When you use this standard deviation program, you enter a list of numbers by typing each number and then pressing the **[RETN]** key. You press **[RETN]** twice after the last number to end the list. The program then prints the average and the standard deviation of the values in the list.

Program

```
10 N=0:SUM=0:SUM 2=0 [RETN]
20 PRINT:PRINT "Enter a list of numbers to check." [RETN]

30 PRINT "Press RETN twice after last number." [RETN]
40 PRINT N+1;:INPUT A$ [RETN]


50 IF A$="" THEN 110 [RETN]


60 X = VAL(A$) [RETN]

70 SUM = SUM+X [RETN]
80 SUM2 = SUM2+X*X [RETN]

90 N = N+1 [RETN]
100 GOTO 40 [RETN]

110 AVERAGE = SUM/N [RETN]
120 PRINT "Average =";AVERAGE [RETN]
130 DEV = SQR((SUM2-SUM*SUM/N)/(N-1))[RETN]

140 PRINT "Standard Deviation =";DEV [RETN]
150 INPUT "Another list (y or n)";A$ [RETN]
160 IF A$="y" OR A$="Y" THEN 10 [RETN]
```

Sets all variables to zero.
Prints a blank line and then prints a prompt.
Prints a prompt.
Prints the count and then records an entry from the keyboard.
Computes the standard deviation if **[RETN]** *is pressed with no number.*
Places the value of the entry into X.
Adds the value to SUM.
Adds the square of value to SUM2.
Raises the entry count by 1.
Returns to line 40 to get the next entry.
Computes the average.
Prints the average.
Computes the standard deviation.
Prints the standard deviation.
Asks if the session is over.
If the session is not over, restarts at line 10.

Sample Run

run [RETN]

```
Enter a list of numbers to check.
Press RETN twice after last number.
 1 ? 22 [RETN]
 2 ? 23 [RETN]
 3 ? 45 [RETN]
 4 ? 34 [RETN]
 5 ? [RETN]
Average = 31
Standard Deviation = 10.80123
Another list (y or n)? n [RETN]
Ok
```

DECISION MAKER

This decision maker program helps you make decisions by prioritizing your choices. The first part of the program asks how many choices there are. You enter the number and then type the name of each choice. Next, the program asks for the number of criteria by which you are judging the choices and for the names of these criteria.

The second part of the program records the rating, from 0 to 100, that you give each criterion. When you are done entering the individual criterion ratings, the program prints the compiled ratings for each choice.

Program

```
10 PRINT:INPUT "How many choices are there";CH [RETN]

20 FOR X = 1 TO CH [RETN]
30 PRINT "      Enter choice";X; [RETN]
40 INPUT A$(X) [RETN]
50 D(X) = 0 [RETN]

60 NEXT X [RETN]

70 INPUT "How many criteria do you have";CR [RETN]
80 FOR X=1 TO CR [RETN]
90 PRINT "      Enter criterion";X; [RETN]
100 INPUT B$(X) [RETN]
110 NEXT X [RETN]

120 PRINT "Rate the choices 1 to 100." [RETN]
130 FOR X=1 TO CH [RETN]
140 PRINT "      Choice:";A$(X) [RETN]
150 FOR Y=1 TO CR [RETN]

160 PRINT "          ";B$(Y); [RETN]
170 INPUT " rating:";C [RETN]
180 D(X) = D(X)+C [RETN]
190 NEXT Y [RETN]

200 NEXT X [RETN]
210 PRINT:PRINT "Compiled Ratings:" [RETN]

220 FOR X=1 TO CH [RETN]
230 PRINT "      ";A$(X);D(X) [RETN]
240 NEXT X [RETN]
250 PRINT:INPUT "Another decision (y or n)";A$ [RETN]

260 IF A$="y" OR A$="Y" THEN 10 [RETN]
```

Prints a blank line and then asks for number of choices.
Steps through each choice.
Prints a prompt.
Records the choices.
Clears the total for the next decision.
Returns to line 20 until all choices are entered.
Asks for number of criteria.
Steps through each criterion.
Prints a prompt.
Records the criterion.
Returns to line 80 until all choices are entered.
Prints instructions for user.
Steps through each choice.
Prints a choice.
Steps through the criteria for each choice.
Prints a criterion.
Records the response.
Adds the ratings.
Returns to print the next criterion.
Returns to print the next choice.
Prints a blank line and then prints a title for the results.
Steps through each choice.
Prints the compiled rating.
Returns to print the next rating.
Prints a blank line and then asks if the session is over.
If the session is not over, restarts at line 10.

continued on next page

Sample Run

run [RETN]

How many choices are there? **3 [RETN]**
    Enter choice 1 ? **ice cream [RETN]**
    Enter choice 2 ? **hot dogs [RETN]**
    Enter choice 3 ? **liver [RETN]**
How many criteria do you have? **2 [RETN]**
    Enter criterion 1 ? **taste [RETN]**
    Enter criterion 2 ? **nutrition [RETN]**
Rate the choices 1 to 100.
    Choice: ice cream
    taste rating: ? **95 [RETN]**
    nutrition rating: ? **45 [RETN]**
    Choice: hot dogs
    taste rating: ? **78 [RETN]**
    nutrition rating: ? **66 [RETN]**
    Choice: liver
    taste rating: ? **12 [RETN]**
    nutrition rating: ? **89 [RETN]**

Compiled Ratings:
    ice cream 140
    hot dogs 144
    liver 101

Another decision (y or n)? **n [RETN]**
Ok

OVER-AND-UNDER DICE GAME

This program plays a game in which you bet on imaginary dice. The program begins by asking you how much you want to bet on each throw of the dice. Next you must guess whether the total number showing on two dice will be over 7, under 7, or even with 7. The program throws the imaginary dice, tells you that you won or lost, and adjusts your winnings (or losses) accordingly.

Program

```
10 PRINT:INPUT "Enter a random number";N [RETN]
20 RANDOMIZE(N) [RETN]
30 INPUT "How much to bet";AM [RETN]
40 PRINT:PRINT "Enter u for under 7,  "; [RETN]
50 INPUT "o for over, e for even";A$ [RETN]
60 D1 = INT(6*RND)+1 [RETN]
70 D2 = INT(6*RND)+1 [RETN]
80 THROW = D1+D2 [RETN]
90 PRINT "Your throw is";THROW; [RETN]
100 IF THROW < 7 AND A$="u" THEN 160 [RETN]
110 IF THROW = 7 AND A$="e" THEN 160 [RETN]
120 IF THROW > 7 AND A$="o" THEN 160 [RETN]
130 TOTAL = TOTAL-AM [RETN]
140 PRINT "and you lose." [RETN]
150 GOTO 180 [RETN]
160 TOTAL = TOTAL+AM [RETN]
170 PRINT "and you win!" [RETN]
180 GAME = GAME+1 [RETN]
190 PRINT "Game";GAME;"total: $";TOTAL [RETN]
200 INPUT "Try again (y or n)";A$ [RETN]
210 IF A$="y" OR A$="Y" THEN 40 [RETN]
```

Prints a blank line and then records a random number entered by the user.
Sets the random number generator.
Records the standard bet.
Prints a blank line and then prints the first part of a prompt.
Completes printing the prompt and then records the player's dice prediction.
Throws the first die.
Throws the second die.
Adds the two together.
Prints the results.
Player wins on under.
Player wins on even.
Player wins on over.
If the player loses, subtracts bet.
Informs player of loss.
Jumps past player's reward.
If the player wins, adds bet to total.
Informs player of win.
Adds one to the game counter.
Prints current "winnings".
Asks if the game is over.
If the game is not over, returns to line 40.

Sample Run

Note that the results from this sample run may not match the results from your sample run because the DOT generates a random number for the throw of the dice.

run [RETN]

Enter a random number? **10 [RETN]**
How much to bet? **100 [RETN]**

Enter u for under 7, o for over, e for even? **o [RETN]**
Your throw is 8 and you win!
Game 1 total: $ 100
Try again (y or n)? **y [RETN]**

continued on next page

```
Enter u for under 7, o for over, e for even? u [RETN]
Your throw is 7 and you lose.
Game 2 total: $ 0
Try again (y or n)? n [RETN]
Ok
```

FACTORS OF A NUMBER

This program determines whether a number is prime or not, that is, whether the number is evenly divisible by another number. If the number you enter is not prime, the program prints the factors that comprise the number.

Program

```
10 FLAG = 0 [RETN]

20 PRINT:INPUT "Enter a number to factor"; N [RETN]

30 FOR X=N-1 TO 2 STEP -1 [RETN]

40 IF N/X=INT(N/X) THEN GOSUB 90 [RETN]
50 NEXT X [RETN]
60 IF FLAG=0 THEN PRINT N;" is prime" [RETN]
70 INPUT "Try another (y or n)";A$ [RETN]
80 IF A$="y" OR A$="Y" THEN 10 ELSE END [RETN]

90 IF FLAG=0 THEN PRINT "Factors:" [RETN]

100 FLAG=1 [RETN]
110 PRINT X [RETN]
120 RETURN [RETN]
```

Zero means the number is prime. The program assumes this at the start.
Prints a blank line, prints a prompt for user input, and then records the number to factor.
Steps through all numbers from N − 1 to 2.
Looks for even division of N.
Tries the next lower number.
Prime has been found.
Asks if session is over.
If the session is not over, restarts at line 10.
This subroutine begins by printing a Factors heading.
Factors exist, so sets flag.
Prints this factor.
Returns to the main program at line 50.

Sample Run

```
run [RETN]

Enter a number to factor? 15 [RETN]
Factors:
 5
 3
Try another (y or n)? y [RETN]

Enter a number to factor? 5 [RETN]
 5  is prime
Try another (y or n)? n [RETN]
Ok
```

NUMBER AVERAGING

This program averages numbers. You simply enter the numbers and the program prints the average.

Program

```
10 COUNT = 1 [RETN]

20 TOTAL = 0 [RETN]
30 PRINT:PRINT "Enter list of numbers to average." [RETN]

40 PRINT "Press RETN twice after last number." [RETN]
50 PRINT:PRINT "Number";COUNT; [RETN]

60 INPUT A$ [RETN]
70 IF A$="" THEN 130 [RETN]

80 TOTAL = TOTAL+VAL(A$) [RETN]

90 AVERAGE = TOTAL/COUNT [RETN]
100 PRINT "Average =";AVERAGE [RETN]
110 COUNT = COUNT+1 [RETN]
120 GOTO 50 [RETN]

130 INPUT "Another try (y or n)";A$ [RETN]
140 IF A$="y" OR A$="Y" THEN 10 [RETN]
```

Sets the count for the first number.
Sets the initial total to zero.
Prints a blank line and then prints the instructions.
Prints more instructions.
Prints a blank line and then prints the current count.
Records the input number.
If no input number, then the program is done.
Adds the input number to the total.
Computes the average.
Prints the results so far.
Adds one to the count.
Goes back to line 50 to get the next number.
Asks if the session is over.
If the session is not over, restarts at line 10.

Sample Run

```
run [RETN]

Enter list of numbers to average.
Press RETN twice after last number.

Number 1? 22 [RETN]
Average = 22

Number 2? 44 [RETN]
Average = 33

Number 3? 57 [RETN]
Average = 41

Number 4? [RETN]
Another try (y or n)? n [RETN]
Ok
```

WALLPAPER COST ESTIMATING

This program determines the cost to wallpaper a room. You enter the number of walls in the room, the height of the ceiling, the length of each wall and the wallpaper cost per foot. You also indicate if you are using wallpaper with an edge pattern that needs to be matched.

This program assumes a $30 base fee for labor and, assuming that the installation rate is 1 foot per minute, adds $10 per hour on a weekday. If you indicate Saturday installation, the program figures the labor at $15 per hour.

Program

```
10 QTY = 0:PRINT [RETN]                                   Sets the initial quantity to zero and prints a blank line.

20 INPUT "How many walls"; WN [RETN]                      Records the number of walls.
30 INPUT "Wall height (inches)";H [RETN]                  Records the ceiling height.
40 FOR X=1 TO WN [RETN]                                   Steps through one wall at a time.

50 PRINT "Length of wall";X; [RETN]                       Prints a prompt for the wall length.

60 INPUT A(X) [RETN]                                      Records the length of each wall.
70 NEXT X [RETN]                                          Returns to line 40 for the next wall.

80 INPUT "Wallpaper cost ( $/ft)";COST [RETN]             Records the wallpaper cost per foot.

90 COST = COST/12 [RETN]                                  Converts cost per foot to cost per inch.

100 INPUT "Patterned-edge paper (y or n)";P$ [RETN]       Records patterned or plain wallpaper.

110 IF P$="y" THEN P$="Y" [RETN]                          Converts "y" to "Y" for line 200.
120 INPUT "Saturday work (y or n)";A$ [RETN]              Records Saturday or weekday work.

130 IF A$="y" THEN A$="Y" [RETN]                          Converts "y" to "Y" for line 270.
140 FOR X=1 TO WN [RETN]                                  Steps through one wall at a time.

150 STRIPS = A(X)/24 [RETN]                               Computes the number of 24-inch strips for each wall.

160 S = INT(STRIPS) [RETN]                                Computes the whole number (removes fractions, if any).

170 IF S <> STRIPS THEN STRIPS = S+1 [RETN]               Computes the next larger whole number if fraction exists.

180 QTY = QTY + STRIPS*H [RETN]                           Sums the strips and computes the total length required in inches.

190 NEXT X [RETN]                                         Returns to line 140 for the next wall.

200 IF P$="Y" THEN QTY=QTY*2 [RETN]                       Doubles the required length if you use paper with patterned edges.

210 FT = INT(QTY/12) [RETN]                               Computes the number of feet required.

220 IN = QTY-INT(QTY/12)*12 [RETN]                        Computes the number of inches to add to feet.
```

Program continued on next page

```
230 PRINT:PRINT "Get";FT;"ft.,";IN;"in. of paper" [RETN]
```
Prints a blank line and then prints the quantity of wallpaper required.

```
240 PRINT "Your material cost will be $";QTY*COST [RETN]
```
Prints the material cost.

```
250 LBR = QTY/720 [RETN]
```
Computes the number of labor hours assuming 1 ft/min at $10/hour.

```
260 PRINT "Labor will require";LBR;"hours" [RETN]
```
Prints the number of labor hours.

```
270 IF A$="Y" THEN LBR=LBR*1.5:PRINT "Saturday"; [RETN]
```
Adjusts the rate for time and a half on Saturday.

```
280 PRINT "Labor will cost $";LBR*10+30 [RETN]
```
Prints the labor cost including the $30 minimum.

```
290 PRINT "Total cost = $";COST*QTY+LBR*10+30 [RETN]
```
Prints the total job cost.

```
300 INPUT "Another try (y or n)";A$ [RETN]
```
Asks if the session is over.

```
310 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
If the session is not over, restarts at line 10.

Sample Run

run [RETN]

```
How many walls? 2 [RETN]
Wall height (inches)? 120 [RETN]
Length of wall 1? 344 [RETN]
Length of wall 2? 409 [RETN]
Wallpaper cost ( $/ft )? .6 [RETN]
Patterned-edge paper (y or n)? n [RETN]
Saturday work (y or n?) n [RETN]

Get 330 ft., 0 in. of paper
Your material cost will be $ 198
Labor will require 5.5 hours
Labor will cost $ 85
Total cost = $ 283
Another try (y or n)? y [RETN]

How many walls? 1 [RETN]
Wall height (inches)? 120 [RETN]
Length of wall 1 ? 284 [RETN]
Wallpaper cost ( $/ft )? .8 [RETN]
Patterned-edge paper (y or n)? y [RETN]
Saturday work (y or n)? y [RETN]
Get 240 ft., 0 in. of paper
Your material cost will be $ 192
Labor will require 4 hours
Saturday labor will cost $ 90
Total cost = $ 282
Another try (y or n)? n [RETN]
Ok
```

HANGMAN

This popular game tests your skill at guessing words when provided with only a few letters. This version of Hangman uses four-letter words (none are offensive) that do not contain the same letter twice.

The object of the game is to guess each letter in the word. If you guess a letter correctly, the letter appears in its correct position in the word. You win when you guess all four letters correctly. If you guess incorrectly, the program prints a sentence saying how much of the man is hanging in the noose. If you guess seven incorrect letters, the program hangs the complete body and you lose.

To enlarge the list of four-letter words, you can add more words by inserting new DATA statements at the end of the program. You must also adjust the number 16 in line 40 to reflect the new number of words in the data list.

Program

```
10 PRINT:INPUT "Enter a random number";RN [RETN]
```
Prints a blank line and then records a random number from user.

```
20 RANDOMIZE(RN) [RETN]
```
Sets the random number generator.

```
30 W = 0: R = 0 [RETN]
```
Sets the wrong and right counters to zero.

```
40 FOR X=1 TO INT(RND*16)+1 [RETN]
```
Selects a random word from the list of 16 words.

```
50 READ A$ [RETN]
```
Reads the selected word from the DATA list below.

```
60 NEXT X [RETN]
```
Returns to line 40 until it reads the selected word.

```
70 ANS$ = " ' "+A$+"'" [RETN]
```
Saves the word to display later if you lose.

```
80 B$ = "----" [RETN]
```
Creates the blank form that the user fills in with correct guesses.

```
90 PRINT "The word is :";B$ [RETN]
```
Prints the form with correct guesses filled in.

```
100 INPUT "Guess a letter";C$ [RETN]
```
Records the guess.

```
110 FLAG = 0 [RETN]
```
Resets the correct letter flag to zero before checking the word.

```
120 FOR X=1 TO 4 [RETN]
```
Steps through all four letters.

```
130 IF MID$(A$,X,1)=C$ THEN GOSUB 220 [RETN]
```
If the guess matches a letter, the program goes to line 220.

```
140 NEXT X [RETN]
```
Returns to 120 to check the next letter in the word.

```
150 IF FLAG=0 THEN GOSUB 280 [RETN]
```
If correct letter flag is still zero, the guess is wrong.

```
160 IF R=4 THEN PRINT ANS$; " YOU WIN!":GOTO 190 [RETN]
```
Signals a win if the player guesses four correct letters.

```
170 IF W=7 THEN 190 [RETN]
```
Signals a loss if the player guesses seven incorrect letters.

```
180 GOTO 90 [RETN]
```
Returns to line 90 for the next guess.

Program continued on next page

```
190 INPUT "Try another word (y or n)";A$ [RETN]
```
Asks if the player wants to try again.

```
200 IF A$="y" OR A$="Y" THEN RESTORE:GOTO 30 [RETN]
```
If so, returns to line 30 for the next game.

```
210 END [RETN]
```
Stops the program if the player does not want to play again.

```
220 MID$(B$,X,1)=C$ [RETN]
```
Begins the correct character subroutine by placing the correct letter in the blank form.

```
230 MID$(A$,X,1)="@" [RETN]
```
Removes the correct letter from the original word so the player cannot select it twice.

```
240 R = R+1 [RETN]
```
Adds one to the counter for the right letters.

```
250 FLAG = 1 [RETN]
```
Sets the correct letter flag to indicate a correct letter.

```
260 PRINT "Correct.  "; [RETN]
```
Prints "Correct.", followed by a space.

```
270 RETURN [RETN]
```
Returns to line 140 to check the next letter position.

```
280 W = W+1 [RETN]
```
Begins the incorrect character subroutine by adding 1 to the counter for wrong letters.

```
290 PRINT "Wrong.  "; [RETN]
```
Prints "Wrong.", followed by a space.

```
300 ON W GOTO 310,320,330,340,350,360,370 [RETN]
```
Selects the proper message for this wrong guess.

```
310 PRINT "Your head is in the noose.":GOTO 390 [RETN]
```
Prints the message for the first wrong letter.

```
320 PRINT "You've got a body now.":GOTO 390 [RETN]
```
Prints the message for the second wrong letter.

```
330 PRINT "Your left leg is showing.":GOTO 390 [RETN]
```
Prints the message for the third wrong letter.

```
340 PRINT "Now you have two legs.":GOTO 390 [RETN]
```
Prints the message for the fourth wrong letter.

```
350 PRINT "Add a left arm now.":GOTO 390 [RETN]
```
Prints the message for the fifth wrong letter.

```
360 PRINT "Right arm, death is near...":GOTO 390 [RETN]
```
Prints the message for the sixth wrong letter.

```
370 PRINT "Sorry, you just hung yourself!" [RETN]
```
Prints the message for the seventh wrong letter.

```
380 PRINT "The word was" ;ANS$;"." [RETN]
```
Prints the answer.

```
390 RETURN [RETN]
```
Returns to line 160 of the main program.

```
400 DATA jamb,glen,blow,bare,yank,zany,taxi,rind [RETN]
```
Stores the words that the program selects in lines 40 to 60.

```
410 DATA oath,nigh,idea,diet,brag,waft,user,hazy [RETN]
```
Stores the words that the program selects in lines 40 to 60.

Sample Run

Note that the results from this sample run may not match the results from your sample run because this program selects words at random.

run [RETN]

Enter a random number? **10 [RETN]**
The word is ----.

continued on next page

Guess a letter? **a [RETN]**
Wrong. Your head is in the noose.
The word is ----.
Guess a letter? **e [RETN]**
Correct. The word is --e-.
Guess a letter? **s [RETN]**
Wrong. You've got a body now.
The word is --e-.
Guess a letter? **t [RETN]**
Wrong. Your left leg is showing.
The word is --e-.
Guess a letter? **n [RETN]**
Correct. The word is --en.
Guess a letter? **l [RETN]**
Correct. The word is -len.
Guess a letter? **g [RETN]**
Correct. 'glen' YOU WIN!
Try another word (y or n)? **n [RETN]**
Ok

LOAN PAYMENTS

If you are considering taking out a loan, you are probably interested in how much each payment will have to be. This program enables you to vary the factors that determine the periodic payment. When you enter the annual interest rate, the number of payments per year, the amount borrowed, and the term of the loan in years, the program prints the payment amount.

Program

```
10 PRINT:INPUT "Annual interest rate";AR [RETN]
```
Prints a blank line and then records the interest rate.

```
20 DAR = AR/100 [RETN]
```
Converts the percentage to a decimal value.

```
30 INPUT "Number of payments per year";NP [RETN]
```
Prints a prompt for the number of payments and then records the number of payments.

```
40 PR = DAR/NP [RETN]
```
Converts the annual interest to periodic interest.

```
50 INPUT "Amount borrowed";AM [RETN]
```
Prints a prompt for the borrowed amount and then records the borrowed amount.

```
60 INPUT "Term of loan in years";YRS [RETN]
```
Prints a prompt for the term of the loan and then records the term of the loan.

```
70 PMT = (DAR*AM/NP)/(1-(PR+1)/\(-NP*YRS))[RETN]
```
Computes the periodic payments.

```
80 PRINT:PRINT "Each payment = $";PMT [RETN]
```
Prints a blank line and then prints the amount of the payments.

```
90 INPUT "Another try (y or n)";A$ [RETN]
```
Asks if the session is over.

```
100 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
Restarts at line 10 if the session is not over.

Sample Run

run [RETN]

Annual interest rate? **14.8 [RETN]**
Number of payments per year? **12 [RETN]**
Amount borrowed? **4000 [RETN]**
Term of loan in years? **4.5 [RETN]**

Each payment = $ 101.8979
Another try (y or n)? **y [RETN]**

Annual interest rate? **14.8 [RETN]**
Number of payments per year? **12 [RETN]**
Amount borrowed? **6000 [RETN]**
Term of loan in years? **3 [RETN]**

Each payment = $ 207.4041
Another try (y or n)? **n [RETN]**
Ok

LOAN PAYOFF

Should you suddenly obtain a lot of money that you don't know what to do with, you may decide to pay off a loan. Using this program, you can calculate how much you would have to pay to close your loan at any point in the term of the loan. You simply enter the monthly payment amount, the annual interest rate, the total number of payments in the loan, and the number of payments already made. The program prints the payoff amount.

Program

```
10 PRINT:INPUT "Monthly payment"; P [RETN]
```
Prints a blank line, prints a prompt for the payment amount, and then records the payment amount.

```
20 INPUT "Annual interest rate"; AR [RETN]
```
Prints a prompt for the interest rate and then records the interest rate.

```
30 DAR = AR/100 [RETN]
```
Computes the decimal annual rate.

```
40 PR = DAR/12 [RETN]
```
Computes the monthly rate.

```
50 INPUT "Total number of payments"; N [RETN]
```
Prints a prompt for the number of payments and then records the number of payments.

```
60 INPUT "Number of past payments made"; PP [RETN]
```
Prints a prompt for past payments and then records the number of past payments.

```
70 PAYOFF = P*((1-(1+PR)/\(PP-N))/PR) [RETN]
```
Computes the loan payoff amount.

```
80 PRINT:PRINT "Payoff amount = $";PAYOFF [RETN]
```
Prints a blank line and then prints the loan payoff amount.

```
90 INPUT "Another try (y or n)";A$ [RETN]
```
Asks if the session is over.

```
100 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
Restarts at line 10 if the session is not over.

Sample Run

run [RETN]

Monthly payment? **215.44 [RETN]**
Annual interest rate? **16 [RETN]**
Total number of payments? **24 [RETN]**
Number of past payments made? **12 [RETN]**

Payoff amount = $ 2374.49
Another try (y or n)? **y [RETN]**

Monthly payment? **215.44 [RETN]**
Annual interest rate? **16 [RETN]**
Total number of payments? **24 [RETN]**
Number of past payments made? **16 [RETN]**

Payoff amount = $ 1624.538
Another try (y or n)? **n [RETN]**
Ok

COSTING REAL ESTATE POINTS

When buying a house with an FHA loan, you may have to pay for points on the mortgage depending on the current bank mortgage rates. This program determines how many points you are paying for and how much these points cost. You simply enter the FHA rate, the current bank rate, the purchase price, and the percentage of the purchase price that you are covering with a down payment.

Program

```
10 PRINT:INPUT "Bank mortgage rate"; BR [RETN]
```
Prints a blank line, prints a prompt for the bank rate, and then records the bank rate.

```
20 INPUT "FHA mortgage rate"; FHA [RETN]
```
Prints a prompt for the FHA rate and then records the FHA rate.

```
30 INPUT "Selling price"; SP [RETN]
```
Prints a prompt for the selling price and then records the selling price.

```
40 INPUT "What percent down payment"; DP [RETN]
```
Prints a prompt for the down payment and then records the down payment percentage.

```
50 DP = DP/100 [RETN]
```
Computes the decimal down payment fraction.

```
60 PTS = (BR-FHA)*4 [RETN]
```
Computes the number of points.

```
70 PTCST = (SP-DP*SP)*PTS*.01 [RETN]
```
Computes the cost of the points.

```
80 PRINT:PRINT "You'll pay $";PTCST;"on"; [RETN]
```
Prints a blank line and then prints the cost of the points.

```
90 PRINT PTS;"points for this loan." [RETN]
```
Prints the number of points.

```
100 PRINT:INPUT "Try another (y or n)";A$ [RETN]
```
Prints a blank line and then asks if the session is over.

```
110 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
Restarts at line 10 if the session is not over.

Sample Run

run [RETN]

Bank mortgage rate? **16.85 [RETN]**
FHA mortgage rate? **15.3 [RETN]**
Selling price? **65000 [RETN]**
What percent down payment? **10 [RETN]**

You'll pay $ 3626.996 on 6.199993 points for this loan.

Try another (y or n)? **y [RETN]**

Bank mortgage rate? **16.85 [RETN]**
FHA mortgage rate? **15.3 [RETN]**
Selling price? **65000 [RETN]**
What percent down payment? **20 [RETN]**

You'll pay $ 3223.996 on 6.199993 points for this loan.

Try another (y or n)? **n [RETN]**
Ok

AGE IN DAYS

Have you ever wondered how many days old you are? This program answers this important question for you. You enter today's date and your birthdate. The program calculates how many days have passed since you were born.

Program

```
10 PRINT:INPUT "Today's date (mm,dd,yy)";M2,D2,Y2 [RETN]
```
Prints a blank line and then records today's date.

```
20 PRINT:INPUT "Birthdate (mm,dd,yy)";M1,D1,Y1 [RETN]
```
Prints a blank line and then records the birthdate.

```
30 M = M1:D = D1:Y = Y1 [RETN]
```
Prepares for entry into subroutine 120.

```
40 GOSUB 120 [RETN]
```
Activates the subroutine.

```
50 N = A [RETN]
```
Saves the results from the subroutine.

```
60 M=M2:D=D2:Y=Y2 [RETN]
```
Prepares for reentry into subroutine 120.

```
70 GOSUB 120 [RETN]
```
Activates the subroutine.

```
80 N = A-N [RETN]
```
Calculates the number of days since birth.

```
90 PRINT:PRINT "You are";N;"days old." [RETN]
```
Prints a blank line and then prints the number of days since birth.

```
100 PRINT:INPUT "Another try (y or n)";A$ [RETN]
```
Prints a blank line and then asks if the session is over.

```
110 IF A$="y" OR A$="Y" THEN 20 ELSE END [RETN]
```
Returns to line 20 if the session is not over.

```
120 RESTORE [RETN]
```
Begins the subroutine by reading the DATA list at the beginning.

```
130 FOR H=1 TO M [RETN]
```
Moves to the right month in the DATA list.

```
140 READ A [RETN]
```
Gets the cumulative number of days from the DATA list.

```
150 NEXT H [RETN]
```
Returns to line 130 for the next month in the DATA list.

```
160 A = A+Y*365+INT(Y/4)+D+1-INT(Y/100)+INT(Y/400) [RETN]
```
Computes an absolute date.

```
170 IF INT(Y/4)<>Y/4 THEN 220 [RETN]
```
Performs the first check for leap years.

```
180 IF Y/400=INT(Y/400) THEN 200 [RETN]
```
Performs the second check for leap years.

```
190 IF Y/100=INT(Y/100) THEN 220 [RETN]
```
Performs the third check for leap years.

```
200 IF M>2 THEN 220 [RETN]
```
Skips the next line if after February.

```
210 A = A-1 [RETN]
```
Adjusts the date if before February.

```
220 RETURN [RETN]
```
Returns to the main program from the subroutine.

```
230 DATA 0,31,59,90,120,151,181 [RETN]
```
Stores the number of days to the first day of the first 7 months.

```
240 DATA 212,243,273,304,334 [RETN]
```
Stores the number of days to the first day of the last 5 months.

Sample Run

run [RETN]

Today's date (mm,dd,yy)? **10,10,82 [RETN]**

Birthdate (mm,dd,yy)? **11,8,49 [RETN]**

You are 12024 days old.

Another try (y or n)? **y [RETN]**

Birthdate (mm,dd,yy)? **2,8,61 [RETN]**

You are 7914 days old.

Another try (y or n)? **y [RETN]**

Birthdate (mm,dd,yy)? **10,28,81 [RETN]**

You are 347 days old.

Another try (y or n)? **n [RETN]**

Ok

SALVAGE VALUE

It's a sad fact that everything depreciates. This program calculates the salvage value of an item at the end of a specified time period. The program asks for the item's original price, the depreciation rate, and the number of years over which the item is to be depreciated. After the program prints the salvage value, it asks if you want to recalculate the value based on another time period.

Program

```
10 PRINT:INPUT "Original price"; OP [RETN]
```
Prints a blank line, prints a prompt for the price, and then records the price.

```
20 INPUT "Depreciation rate (%)"; DR [RETN]
```
Prints a prompt for the depreciation rate and then records the depreciation rate.

```
30 DDR = DR/100 [RETN]
```
Computes the decimal depreciation rate.

```
40 INPUT "Age in years"; YRS [RETN]
```
Prints a prompt for the age and then records the age.

```
50 VALUE = OP*(1-DDR)^YRS [RETN]
```
Computes the salvage value.

```
60 PRINT:PRINT "Salvage value = $";VALUE [RETN]
```
Prints a blank line and then prints the salvage value.

```
70 PRINT:INPUT "Try another aging (y or n)";A$ [RETN]
```
Prints a blank line and then asks if you want different aging.

```
80 IF A$="y" OR A$="Y" THEN 40 [RETN]
```
Returns to line 40 for next age calculation.

```
90 PRINT:INPUT "Try another item (y or n)";A$ [RETN]
```
Prints a blank line and then asks if the session is over.

```
100 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
Restarts at line 10 if the session is not over.

Sample Run

run [RETN]

```
Original price? 5000 [RETN]
Depreciation rate (%)? 25 [RETN]
Age in years? 4 [RETN]

Salvage value = $ 1582.031

Try another aging (y or n)? y [RETN]
Age in years? 5 [RETN]

Salvage value = $ 1186.524

Try another aging (y or n)? y [RETN]
Age in years? 6 [RETN]

Salvage value = $ 889.8926

Try another aging (y or n)? n [RETN]

Try another item (y or n)? n [RETN]
Ok
```

EFFECTIVE INTEREST RATE

The many investment plans available can make investment decisions difficult. This program helps you to compare interest rates between different investment plans. You enter the amount invested, the number of years that the amount is invested, and the final investment value. The program calculates and prints the effective interest rate for the investment.

Program

```
10 PRINT:INPUT "Amount invested"; AM [RETN]
```
Prints a blank line, prints a prompt for the amount invested, and then records the amount.

```
20 INPUT "Number of years invested"; YRS [RETN]
```
Prints a prompt for the number of years invested and then records the number of years invested.

```
30 PRINT "Value after";YRS;"years"; [RETN]
```
Prints a prompt for the final value.

```
40 INPUT VALUE [RETN]
```
Records the final value.

```
50 EIR = ((VALUE/AM)^(1/YRS)-1)*100 [RETN]
```
Computes the effective interest.

```
60 PRINT:PRINT "Effective interest rate =";EIR;"%" [RETN]
```
Prints a blank line and then prints the effective interest rate.

```
70 PRINT:INPUT "Another try (y or n)";A$ [RETN]
```
Prints a blank line and then asks if the session is over.

```
80 IF A$="y" OR A$="Y" THEN 10 [RETN]
```
Restarts at line 10 if the session is not over.

Sample Run

run [RETN]

Amount invested? **1000 [RETN]**
Number of years invested? **5 [RETN]**
Value after 5 years? **1420.30 [RETN]**

Effective interest rate = 7.269442 %

Another try (y or n)? **y [RETN]**

Amount invested? **20000 [RETN]**
Number of years invested? **2 [RETN]**
Value after 2 years? **24032 [RETN]**

Effective interest rate = 9.617508 %

Another try (y or n)? **n [RETN]**
Ok

Section 5
AVAILABLE BASIC TEXTBOOKS

Section 5
AVAILABLE BASIC TEXTBOOKS

Many sources are available for more information about the BASIC language. The **Microsoft BASIC** manual that comes with the DOT is a good technical reference for your version of the BASIC language program. However, you may want to start with a beginner's BASIC book before tackling the technical details.

This section provides a list of easy-to-read books that will help you learn the BASIC language. You can obtain these books at most computer stores or through the mail from advertisements in computer publications. Note that the books listed here are written specifically for users of Microsoft BASIC.

You might also investigate BASIC classes in your area. Many community colleges and computer stores offer courses in BASIC.

Microsoft BASIC
Ken Knecht
IBSN 0-88056-056-8
Dilithium Press, 1982
PO Box 606
Beaverton, OR 97005

Practical BASIC Programs
Lon Poole
IBSN 0-931988-38-1
Osborne McGraw-Hill, 1980
630 Bancroft Way
Berkeley, CA 94710

Some Common BASIC Programs, Third Edition
Lon Poole and Mary Borchers
IBSN 0-931988-06-3
Osborne McGraw-Hill, 1979
630 Bancroft Way
Berkeley, CA 94710

50 BASIC Exercises
J.P. Lamoitier
IBSN 0-89588-056-3
Sybex Inc., 1981
2344 Sixth St.
Berkeley, CA 94710

BASIC Business Software, Second Edition
E.G. Brooner
IBSN 0-672-21751-1
Howard W. Sams & Co., Inc., 1981
4300 West 62nd Street
Indianapolis, IN 46268

BASIC Programming Primer
Mitchell Waite and Michael Pardee
IBSN 0-672-21586-1
Howard W. Sams & Co., Inc., 1978
4300 West 62nd Street
Indianapolis, IN 46268

BASIC Handbook
David A. Liem
IBSN 0-932760-05-8
CompuSoft Publishing Division, 1981
CompuSoft, Inc.
San Diego, CA 92119

Appendix A
GLOSSARY

Application program—Software designed for a specific end use, such as accounts payable or word processing.

Backup copy—A copy of a diskette to ensure against loss of valuable programs, text, or data in case of a power failure, a system failure, or an operator error.

BASIC—An acronym for Beginner's All-purpose Symbolic Instruction Code. BASIC is the programming language used with most personal computers.

Command—An instruction that you use to tell the system to do something.

Control sequence—The act of pressing and holding down the key labeled CTRL and then pressing another character.

Current line—The line containing the cursor. The DOT accepts commands only from the current line.

Cursor—A symbol, such as a blinking underscore, that appears on the screen to indicate where the next character you type will appear.

Diagnostic program—Software that detects and isolates hardware malfunctions.

Directory—A listing of file names stored on a particular diskette.

Disk drive—A mechanical device which stores and retrieves information from diskettes.

Diskette—A magnetic recording medium which stores information for fast retrieval.

Disk format—The way the operating system divides each diskette into concentric rings (tracks) and into preshaped sections (sectors).

Editing—The process used to create and correct information in files stored on diskettes.

File—A collection of information kept together under a unique name.

File name extension—A suffix for a file name that indicates the nature of the file.

Function keys—Specific keys on the keyboard which enable you to issue one or more special commands with a single keystroke.

GOTO—A BASIC keyword that indicates which line in the program to read next, rather than the next numerical line.

Hardware—The mechanical or "hard" components of a microcomputer system in contrast to the "soft" or programming components.

INPUT—A BASIC keyword that prints a question mark on the screen and waits for data from the user. This question mark indicates that the program requires you to enter data before it can continue.

Input data—Information that you enter into a computer for processing.

Keyword—A word in a programming language that defines an operation to be performed.

LIST—A BASIC keyword that displays specified lines of the program on the screen.

LLIST—A BASIC keyword that lists specified lines on the optional printer.

LPRINT—A BASIC keyword that directs the output to the printer.

Memory—The component of the computer that stores information for immediate use. The system stores less urgent information on diskette until needed.

Menu—A list of alternative actions that the system supplies. You decide what operation the system performs next.

MS-DOS—The operating system which controls the operation of the DOT hardware and performs file maintenance.

Operating system—An organized collection of programs that translate your instructions so that the DOT can process them.

PRINT—A BASIC keyword that produces the output on the screen.

Program—A list of instructions that the computer follows to perform a task.

Prompt—A message from the system which informs you that a particular entry is required.

REM—A BASIC keyword enabling you to enter comments that describe or annotate the program. REM stands for "remark".

Standard deviation—The amount of variation in a group of numbers. Standard deviation is also called the root mean square deviation.

System files—Files containing the MS-DOS operating system programs. You must copy these files onto any diskette that contains programs requiring access to MS-DOS.

User files—Files that you create while using a program. For example, a word processing program produces user text files, a spreadsheet program produces user data files, and BASIC produces user program files.

Variable name—An alphanumeric representation of a quantity that can assume any of a given set of values.

Video display unit—The television screen or monitor of a computer or word processor. The video display unit (VDU) is also called the screen or the CRT (cathode ray tube).

Wild card characters—Characters that can replace the actual characters in an original file name. A question mark can replace any single character in a file name and an asterisk can replace an entire group of characters.

To help us to provide you with the best manuals possible, please make your comments and suggestions concerning this manual on the form below. Then detach, fold, tape closed and mail to us. All comments and suggestions become the property of Computer Devices, Inc. Your cooperation is greatly appreciated.

Fold

BUSINESS AND FUN WITH BASIC:
A Sampler of BASIC Programs for the DOT™
0008-14613

Comments:

Cut along dotted line

Fold

Name: _____ Date: _____

Position: _____ Dept: _____

Organization: _____ Phone: _____

Address: _____

(Please tape; postal regulations prohibit the use of staples.)

Fold

No postage
necessary if
mailed in the
United States

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO.200A BURLINGTON, MA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTER DEVICES

25 North Avenue
Burlington, MA 01803

Cut along dotted line

Fold

Computer Devices, Inc., 25 North Avenue, Burlington, Massachusetts 01803

0008-14613/3-83